

LIFO-Backpressure Achieves Near Optimal Utility-Delay Tradeoff

Longbo Huang, Scott Moeller, Michael J. Neely, Bhaskar Krishnamachari

Abstract—There has been considerable recent work developing a new stochastic network utility maximization framework using Backpressure algorithms, also known as MaxWeight. A key open problem has been the development of utility-optimal algorithms that are also delay efficient. In this paper, we show that the Backpressure algorithm, when combined with the LIFO queueing discipline (called LIFO-Backpressure), is able to achieve a utility that is within $O(1/V)$ of the optimal value for any scalar $V \geq 1$, while maintaining an average delay of $O([\log(V)]^2)$ for all but a tiny fraction of the network traffic. This result holds for general stochastic network optimization problems and general Markovian dynamics. Remarkably, the performance of LIFO-Backpressure can be achieved by simply changing the queueing discipline; it requires no other modifications of the original Backpressure algorithm. We validate the results through empirical measurements from a sensor network testbed, which show good match between theory and practice.

Index Terms—Queueing, Dynamic Control, LIFO scheduling, Lyapunov analysis, Stochastic Optimization

I. INTRODUCTION

Recent developments in stochastic network optimization theory have yielded a very general framework that solves a large class of networking problems of the following form: We are given a discrete time stochastic network. The network state, which describes current realization of the underlying network randomness, such as the network channel condition, is time varying according to some probability law. A network controller performs some action based on the observed network state at every time slot. The chosen action incurs a cost,¹ but also serves some amount of traffic and possibly generates new traffic for the network. This traffic causes congestion, and thus leads to backlogs at nodes in the network. The goal of the controller is to minimize its time average cost subject to the constraint that the time average total backlog in the network be kept finite.

This general setting models a large class of networking problems ranging from traffic routing [1], flow utility maximization [2], network pricing [3] to cognitive radio applications [4]. Also, many techniques have also been applied to this problem (see [5] for a survey). Among the approaches

that have been adopted, the family of Backpressure algorithms [6] are recently receiving much attention due to their provable performance guarantees, robustness to stochastic network conditions and, most importantly, their ability to achieve the desired performance *without requiring any statistical knowledge* of the underlying randomness in the network.

Most prior performance results for Backpressure are given in the following $[O(1/V), O(V)]$ utility-delay tradeoff form [6]: Backpressure is able to achieve a utility that is within $O(1/V)$ of the optimal utility for any scalar $V \geq 1$, while guaranteeing a average network delay that is $O(V)$. Although these results provide strong theoretical guarantees for the algorithms, the network delay can actually be unsatisfying when we achieve a utility that is very close to the optimal, i.e., when V is large.

There have been previous works trying to develop algorithms that can achieve better utility-delay tradeoffs. Previous works [7] and [8] show improved tradeoffs are possible for single-hop networks with certain structure, and develops optimal $[O(1/V), O(\log(V))]$ and $[O(1/V), O(\sqrt{V})]$ utility-delay tradeoffs. However, the algorithms are different from basic Backpressure and require knowledge of an “epsilon” parameter that measures distance to a performance region boundary. Work [9] uses a completely different analytical technique to show that similar poly-logarithmic tradeoffs, i.e., $[O(1/V), O([\log(V)]^2)]$, are possible by carefully modifying the actions taken by the basic Backpressure algorithms. However, the algorithm requires a pre-determined learning phase, which adds additional complexity to the implementation. The current work, following the line of analysis in [9], instead shows that similar poly-logarithmic tradeoffs, i.e., $[O(1/V), O([\log(V)]^2)]$, can be achieved by the *original* Backpressure algorithm by simply modifying the service discipline from First-in-First-Out (FIFO) to Last-In-First-Out (LIFO) (called LIFO-Backpressure below). This is a remarkable feature that distinguishes LIFO-Backpressure from previous algorithms in [7] [8] [9], and provides a deeper understanding of backpressure itself, and the role of queue backlogs as Lagrange multipliers (see also [2] [9]). However, this performance improvement is not for free: We must *drop a small fraction of packets* in order to dramatically improve delay for the remaining ones. We prove that as the V parameter is increased, the fraction of dropped packets quickly converges to zero, while maintaining $O(1/V)$ close-to-optimal utility and $O([\log(V)]^2)$ average backlog. This provides an analytical justification for experimental observations in [10] that shows a related LIFO-Backpressure rule serves up to 98% of the traffic with delay that is improved by 2 orders of magnitude.

Longbo Huang, Scott Moeller, Michael J. Neely, and Bhaskar Krishnamachari (emails: {longbohu, smoeller, mjneely, bkrishna}@usc.edu) are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA.

This material is supported in part under one or more of the following grants: DARPA IT-MANET W911NF-07-0028, NSF CAREER CCF-0747525, and continuing through participation in the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory.

¹Since cost minimization is mathematically equivalent to utility maximization, below we will use cost and utility interchangeably

LIFO-Backpressure was proposed in recent empirical work [10]. The authors developed a practical implementation of backpressure routing and showed experimentally that applying LIFO queuing discipline drastically improves average packet delay, but did not provide theoretical guarantees. Another notable recent work providing an alternative delay solution is [11], which describes a novel backpressure-based per-packet randomized routing framework that runs atop the shadow queue structure of [12] while minimizing hop count as explored in [13]. Their techniques reduce delay drastically and eliminates the per-destination queue complexity, but does not provide $O([\log(V)]^2)$ average delay guarantees.

Our analysis of the delay performance of LIFO-Backpressure is based on the recent “exponential attraction” result developed in [9]. The proof idea can be intuitively explained by Fig. 1, which depicts a simulated backlog process of a single queue system with unit packet size under Backpressure. The left figure demonstrates the “exponential attraction”

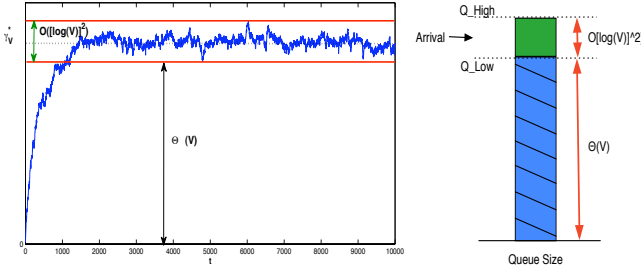


Fig. 1. The LIFO-Backpressure Idea

result in [9], which states that queue sizes under Backpressure deviate from some fixed point with probability that decreases exponentially in the deviation distance. Hence the queue size will mostly fluctuate within the interval $[Q_{Low}, Q_{High}]$ which can be shown to be of $O([\log(V)]^2)$ size. This result holds under both FIFO and LIFO, as they result in the same queue process. Now suppose LIFO is used in this queue. Then from the right figure, we see that most of the packets will arrive at the queue when the queue size is between Q_{Low} and Q_{High} , and these new packets will always be placed on the top of the queue due to the LIFO discipline. Most packets thus enter and leave the queue when the queue size is between Q_{Low} and Q_{High} . Therefore, these packets “see” a queue with average size no more than $Q_{High} - Q_{Low} = O([\log(V)]^2)$. Now let λ be the packet arrival rate into the queue, and let $\tilde{\lambda}$ be the arrival rate of packets entering when the queue size is in $[Q_{Low}, Q_{High}]$ and that eventually depart. Because packets always occupy the same buffer slot under LIFO, we see that these packets can occupy at most $Q_{High} - Q_{Low} + \delta_{max}$ buffer slots, ranging from Q_{Low} to $Q_{High} + \delta_{max}$, where $\delta_{max} = \Theta(1)$ is the maximum number of packets that can enter the queue at any time. We can now apply Little’s Theorem [14] to the buffer slots in the interval $[Q_{Low}, Q_{High} + \delta_{max}]$, and we see that average delay for these packets that arrive when the queue size is in $[Q_{Low}, Q_{High}]$ satisfies:

$$D \leq \frac{Q_{High} - Q_{Low} + \delta_{max}}{\tilde{\lambda}} = \frac{O([\log(V)]^2)}{\tilde{\lambda}}. \quad (1)$$

Finally, the exponential attraction result implies that $\lambda \approx \tilde{\lambda}$. Hence for almost all packets entering the queue, the average delay is $D = O([\log(V)]^2/\lambda)$.

This paper is organized as follows. In Section II, we set up our notations. We then present our system model in Section III. We provide an example of our network in Section IV. We review the Backpressure algorithm in Section V. The delay performance of LIFO-Backpressure is presented in Section VI. Simulation results are presented in Section VII. We then also present experimental testbed results in Section VIII. Finally, we comment on optimizing a function of time averages in Section IX.

II. NOTATIONS

Here we first set up the notations used in this paper: \mathbb{R} represents the set of real numbers. \mathbb{R}_+ (or \mathbb{R}_-) denotes the set of nonnegative (or non-positive) real numbers. \mathbb{R}^n (or \mathbb{R}_+^n) is the set of n dimensional *column* vectors, with each element being in \mathbb{R} (or \mathbb{R}_+). **bold** symbols \mathbf{a} and \mathbf{a}^T represent *column* vector and its transpose. $\mathbf{a} \geq \mathbf{b}$ means vector \mathbf{a} is entrywise no less than vector \mathbf{b} . $\|\mathbf{a} - \mathbf{b}\|$ is the Euclidean distance of \mathbf{a} and \mathbf{b} . $\mathbf{0}$ and $\mathbf{1}$ denote column vector with all elements being 0 and 1. $[a]^+ = \max[a, 0]$ and $\log(\cdot)$ is the natural log.

III. SYSTEM MODEL

In this section, we specify the general network model we use. We consider a network controller that operates a network with the goal of minimizing the time average cost, subject to the queue stability constraint. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, \dots\}$. We assume there are $r \geq 1$ queues in the network.

A. Network State

In every slot t , we use $S(t)$ to denote the current network state, which indicates the current network parameters, such as a vector of channel conditions for each link, or a collection of other relevant information about the current network channels and arrivals. We assume that $S(t)$ evolves according a finite state irreducible and aperiodic Markov chain, with a total of M different random network states denoted as $\mathcal{S} = \{s_1, s_2, \dots, s_M\}$. Let π_{s_i} denote the steady state probability of being in state s_i . It is easy to see in this case that $\pi_{s_i} > 0$ for all s_i . The network controller can observe $S(t)$ at the beginning of every slot t , but the π_{s_i} and transition probabilities are not necessarily known.

B. The Cost, Traffic, and Service

At each time t , after observing $S(t) = s_i$, the controller chooses an action $x(t)$ from a set $\mathcal{X}^{(s_i)}$, i.e., $x(t) = x^{(s_i)}$ for some $x^{(s_i)} \in \mathcal{X}^{(s_i)}$. The set $\mathcal{X}^{(s_i)}$ is called the feasible action set for network state s_i and is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. The cost, traffic, and service generated by the chosen action $x(t) = x^{(s_i)}$ are as follows:

- (a) The chosen action has an associated cost given by the cost function $f(t) = f(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$ (or $\mathcal{X}^{(s_i)} \mapsto \mathbb{R}_-$ in reward maximization problems);

- (b) The amount of traffic generated by the action to queue j is determined by the traffic function $A_j(t) = A_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets;
- (c) The amount of service allocated to queue j is given by the rate function $\mu_j(t) = \mu_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets;

Note that $A_j(t)$ includes both the exogenous arrivals from outside the network to queue j , and the endogenous arrivals from other queues, i.e., the transmitted packets from other queues, to queue j . We assume the functions $f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are continuous, time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{max} \in (0, \infty)$ for all s_i, j , and they are known to the network operator. We also assume that there exists a set of actions $\{x^{(s_i)k}\}_{k=1,2,\dots,\infty}^{i=1,\dots,M}$ with $x^{(s_i)k} \in \mathcal{X}^{(s_i)}$ and some variables $\vartheta_k^{(s_i)} \geq 0$ for all s_i and k with $\sum_k \vartheta_k^{(s_i)} = 1$ for all s_i , such that

$$\sum_{s_i} \pi_{s_i} \left\{ \sum_k \vartheta_k^{(s_i)} [A_j(s_i, x^{(s_i)k}) - \mu_j(s_i, x^{(s_i)k})] \right\} \leq -\eta, \quad (2)$$

for some $\eta > 0$ for all j . That is, the stability constraints are feasible with η -slackness. Thus, there exists a stationary randomized policy that stabilizes all queues (where $\vartheta_k^{(s_i)}$ represents the probability of choosing action $x^{(s_i)k}$ when $S(t) = s_i$) [6].

C. Queueing, Average Cost, and the Stochastic Problem

Let $\mathbf{q}(t) = (q_1(t), \dots, q_r(t))^T \in \mathbb{R}_+^r$, $t = 0, 1, 2, \dots$ be the queue backlog vector process of the network, in units of packets. We assume the following queueing dynamics:

$$q_j(t+1) = \max[q_j(t) - \mu_j(t), 0] + A_j(t) \quad \forall j, \quad (3)$$

and $\mathbf{q}(0) = \mathbf{0}$. By using (3), we assume that when a queue does not have enough packets to send, null packets are transmitted. In this paper, we adopt the following notion of queue stability:

$$\mathbb{E} \left\{ \sum_{j=1}^r q_j \right\} \triangleq \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j=1}^r \mathbb{E} \{ q_j(\tau) \} < \infty. \quad (4)$$

We also use f_{av}^Π to denote the time average cost induced by an action-choosing policy Π , defined as:

$$f_{av}^\Pi \triangleq \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E} \{ f^\Pi(\tau) \}, \quad (5)$$

where $f_{av}^\Pi(\tau)$ is the cost incurred at time τ by policy Π . We call an action-choosing policy *feasible* if at every time slot t it only chooses actions from the feasible action set $\mathcal{X}^{(S(t))}$. We then call a feasible action-choosing policy under which (4) holds a *stable* policy, and use f_{av}^* to denote the optimal time average cost over all stable policies. In every slot, the network controller observes the current network state and chooses a control action, with the goal of minimizing the time average cost subject to network stability. This goal can be mathematically stated as: **(P1)** $\min : f_{av}^\Pi, s.t. (4)$. In the following, we will refer to **(P1)** as the *stochastic problem*.

Note that in some network optimization problems, e.g., [15], the objective of the network controller is to optimize a function of a time average metric. In this case, we see

that the Backpressure algorithm and the deterministic problem presented in the next section can similarly be constructed, but will be slightly different. We will discuss these problems in Section IX.

IV. AN EXAMPLE OF OUR MODEL

Here we provide an example to illustrate our model. Consider the 2-queue network in Fig. 2. In every slot, the network operator decides whether or not to allocate one unit of power to serve packets at each queue, so as to support all arriving traffic, i.e., maintain queue stability, with minimum energy expenditure. We assume the network state $S(t)$, which is the quadruple $(R_1(t), R_2(t), CH_1(t), CH_2(t))$, evolves according to the finite state Markov chain with three states $s_1 = (1, 1, G, B)$, $s_2 = (1, 1, G, G)$, and $s_3 = (0, 0, B, G)$. Here $R_i(t)$ denotes the number of exogenous packet arrivals to queue i at time t , and $CH_i(t)$ is the state of channel i . $R_i(t) = x$ implies that there are x number of packets arriving at queue i at time t . $CH_i(t) = G/B$ means that channel i has a ‘‘Good’’ or ‘‘Bad’’ state. When a link’s channel state is ‘‘Good’’, one unit of power can serve 2 packets over the link, otherwise it can only serve one. We assume power can be allocated to both channels without affecting each other.

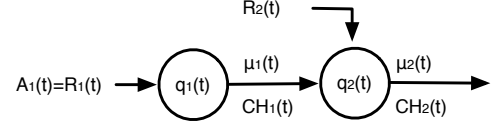


Fig. 2. A two queue tandem example.

In this case, we see that there are three possible network states. At each state s_i , the action $x^{(s_i)}$ is a pair (x_1, x_2) , with x_i being the amount of energy spent at queue i , and $(x_1, x_2) \in \mathcal{X}^{(s_i)} = \{0/1, 0/1\}$. The cost function is $f(s_i, x^{(s_i)}) = x_1 + x_2$, for all s_i . The network states, the traffic functions, and the service rate functions are summarized in Fig. 3. Note here $A_1(t) = R_1(t)$ is part of $S(t)$ and is independent of $x^{(s_i)}$; while $A_2(t) = \mu_1(t) + R_2(t)$ hence depends on $x^{(s_i)}$. Also note that $A_2(t)$ equals $\mu_1(t) + R_2(t)$ instead of $\min[\mu_1(t), q_1(t)] + R_2(t)$ due to our idle fill assumption in Section III-C.

| $S(t)$ | $R_1(t)$ | $R_2(t)$ | $CH_1(t)$ | $CH_2(t)$ | $A_1(t)$ | $A_2(t)$ | $\mu_1(t)$ | $\mu_2(t)$ |
|--------|----------|----------|-----------|-----------|----------|------------|------------|------------|
| s_1 | 1 | 1 | G | B | 1 | $2x_1 + 1$ | $2x_1$ | x_2 |
| s_2 | 1 | 1 | G | G | 1 | $2x_1 + 1$ | $2x_1$ | $2x_2$ |
| s_3 | 0 | 0 | B | G | 0 | x_1 | x_1 | $2x_2$ |

Fig. 3. The traffic and service functions under different states.

V. BACKPRESSURE AND THE DETERMINISTIC PROBLEM

In this section, we first review the Backpressure algorithm [6] for solving the stochastic problem. Then we define the *deterministic problem* and its dual. We first recall the Backpressure algorithm for utility optimization problems [6].

Backpressure: At every time slot t , observe the current network state $S(t)$ and the backlog $\mathbf{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\begin{aligned} \max : \quad & -Vf(s_i, x) + \sum_{j=1}^r q_j(t) [\mu_j(s_i, x) - A_j(s_i, x)] \quad (6) \\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \end{aligned}$$

Depending on the problem structure, (6) can usually be decomposed into separate parts that are easier to solve, e.g., [3], [4]. Also, when the network state process $S(t)$ is i.i.d., it has been shown in [6] that,

$$f_{av}^{BP} = f_{av}^* + O(1/V), \quad \bar{q}^{BP} = O(V), \quad (7)$$

where f_{av}^{BP} and \bar{q}^{BP} are the expected average cost and the expected average network backlog size under Backpressure, respectively. When $S(t)$ is Markovian, [3] and [4] show that Backpressure achieves an $[O(\log(V)/V), O(V)]$ utility-delay tradeoff if the queue sizes are deterministically upper bounded by $\Theta(V)$ for all time. Without this deterministic backlog bound, it has recently been shown that Backpressure achieves an $[O(\epsilon + \frac{T_\epsilon}{V}), O(V)]$ tradeoff under Markovian $S(t)$, with ϵ and T_ϵ representing the proximity to the optimal utility value and the ‘‘convergence time’’ of the Backpressure algorithm to that proximity [16]. However, there has not been any formal proof that shows the exact $[O(1/V), O(V)]$ utility-delay tradeoff of Backpressure under a Markovian $S(t)$.

We also recall the *deterministic problem* defined in [9]:

$$\begin{aligned} \min : \quad & \mathcal{F}(\mathbf{x}) \triangleq V \sum_{s_i} \pi_{s_i} f(s_i, x^{(s_i)}) \quad (8) \\ \text{s.t.} \quad & \mathcal{A}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \\ & \leq \mathcal{B}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}), \quad \forall j, \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall i = 1, 2, \dots, M, \end{aligned}$$

where π_{s_i} corresponds to the steady state probability of $S(t) = s_i$ and $\mathbf{x} = (x^{(s_1)}, \dots, x^{(s_M)})^T$. The dual problem of (8) can be obtained as follows:

$$\max : \quad g(\gamma), \quad \text{s.t.} \quad \gamma \succeq \mathbf{0}, \quad (9)$$

where $g(\gamma)$ is called the dual function and is defined as:

$$\begin{aligned} g(\gamma) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i} \pi_{s_i} \left\{ Vf(s_i, x^{(s_i)}) \right. \\ \left. + \sum_j \gamma_j [A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})] \right\}. \end{aligned} \quad (10)$$

Here $\gamma = (\gamma_1, \dots, \gamma_r)^T$ is the *Lagrange multiplier* of (8). It is well known that $g(\gamma)$ in (10) is concave in the vector γ , and hence the problem (9) can usually be solved efficiently, particularly when cost functions and rate functions are separable over different network components. Below, we use $\gamma_V^* = (\gamma_{V1}^*, \gamma_{V2}^*, \dots, \gamma_{Vr}^*)^T$ to denote an optimal solution of the problem (9) with the corresponding V .

VI. PERFORMANCE OF LIFO BACKPRESSURE

In this section, we analyze the performance of Backpressure with the LIFO queueing discipline (called LIFO-Backpressure). The idea of using LIFO under Backpressure is first proposed in [10], although they did not provide any

theoretical performance guarantee. We will show, under some mild conditions (to be stated in Theorem 3), that under LIFO-Backpressure, the time average delay for almost all packets entering the network is $O([\log(V)]^2)$ when the utility is pushed to within $O(1/V)$ of the optimal value. Note that the implementation complexity of LIFO-Backpressure is the same as the original Backpressure, and LIFO-Backpressure only requires the knowledge of the instantaneous network condition. This is a remarkable feature that distinguishes it from the previous algorithms achieving similar poly-logarithmic tradeoffs in the i.i.d. case, e.g., [7] [8] [9], which all require knowledge of some implicit network parameters other than the instant network state. Below, we first provide a simple example to demonstrate the need for careful treatment of the usage of LIFO in Backpressure algorithms, and then present a modified Little’s theorem that will be used for our proof.

A. A simple example on the LIFO delay

Consider a slotted system where two packets arrive at time 0, and one packet periodically arrives every slot thereafter (at times 1, 2, 3, ...). The system is initially empty and can serve exactly one packet per slot. The arrival rate λ is clearly 1 packet/slot (so that $\lambda = 1$). Further, under either FIFO or LIFO service, there are always 2 packets in the system, so $\bar{Q} = 2$.

Under FIFO service, the first packet has a delay of 1 and all packets thereafter have a delay of 2:

$$W_1^{FIFO} = 1, \quad W_i^{FIFO} = 2 \quad \forall i \in \{2, 3, 4, \dots\},$$

where W_i^{FIFO} is the delay of the i^{th} packet under FIFO (W_i^{LIFO} is similarly defined for LIFO). We thus have:

$$\bar{W}^{FIFO} \triangleq \lim_{K \rightarrow \infty} \frac{1}{K} \sum_{i=1}^K W_i^{FIFO} = 2.$$

Thus, $\lambda \bar{W}^{FIFO} = 1 \times 2 = 2$, $\bar{Q} = 2$, and so $\lambda \bar{W}^{FIFO} = \bar{Q}$ indeed holds.

Now consider the same system under LIFO service. We still have $\lambda = 1$, $\bar{Q} = 2$. However, in this case the first packet never departs, while all other packets have a delay equal to 1 slot:

$$W_1^{LIFO} = \infty, \quad W_i^{LIFO} = 1 \quad \forall i \in \{2, 3, 4, \dots\}.$$

Thus, for all integers $K > 0$:

$$\frac{1}{K} \sum_{i=1}^K W_i^{LIFO} = \infty.$$

and so $\bar{W}^{LIFO} = \infty$. Clearly $\lambda \bar{W}^{LIFO} \neq \bar{Q}$. On the other hand, if we ignore the one packet with infinite delay, we note that all other packets get a delay of 1 (exactly half the delay in the FIFO system). Thus, in this example, LIFO service significantly improves delay for all but the first packet.

For the above LIFO example, it is interesting to note that if we define \tilde{Q} and \tilde{W} as the average backlog and delay associated only with those packets that eventually depart, then we have $\tilde{Q} = 1$, $\tilde{W} = 1$, and the equation $\lambda \tilde{W} = \tilde{Q}$ indeed holds. This motivates the theorem in the next subsection, which considers a time average only over those packets that eventually depart.

B. A Modified Little's Theorem for LIFO systems

We now present the modified Little's theorem. Let \mathcal{B} represent a finite set of buffer locations for a LIFO queueing system. Let $N(t)$ be the number of arrivals that use a buffer location within set \mathcal{B} up to time t . Let $D(t)$ be the number of departures from a buffer location within the set \mathcal{B} up to time t . Let W_i be the delay of the i th job to depart from the set \mathcal{B} . Define \bar{W} as the limsup average delay *considering only those jobs that depart*:

$$\bar{W} \triangleq \limsup_{t \rightarrow \infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i.$$

We then have the following theorem:

Theorem 1: Suppose there is a constant $\lambda_{\min} > 0$ such that with probability 1:

$$\liminf_{t \rightarrow \infty} \frac{N(t)}{t} \geq \lambda_{\min},$$

Further suppose that $\lim_{t \rightarrow \infty} D(t) = \infty$ with probability 1 (so the number of departures is infinite). Then the average delay \bar{W} satisfies:

$$\bar{W} \triangleq \limsup_{t \rightarrow \infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}|/\lambda_{\min},$$

where $|\mathcal{B}|$ is the size of the finite set \mathcal{B} .

Proof: See Appendix A. ■

C. LIFO-Backpressure Proof

We now provide the analysis of LIFO-Backpressure. To prove our result, we first have the following theorem, which is the first to show that Backpressure (with either FIFO or LIFO) achieves the exact $[O(1/V), O(V)]$ utility-delay trade-off under a Markovian network state process. It generalizes the $[O(1/V), O(V)]$ performance result of Backpressure in the i.i.d. case in [6].

Theorem 2: Suppose $S(t)$ is a finite state irreducible and aperiodic Markov chain² and condition (2) holds, Backpressure (with either FIFO or LIFO) achieves the following:

$$f_{av}^{BP} = f_{av}^* + O(1/V), \quad \bar{q}^{BP} = O(V), \quad (11)$$

where f_{av}^{BP} and \bar{q}^{BP} are the expected time average cost and backlog under Backpressure.

Proof: See [17]. ■

Theorem 2 thus shows that LIFO-Backpressure guarantees an average backlog of $O(V)$ when pushing the utility to within $O(1/V)$ of the optimal value. We now consider the delay performance of LIFO-Backpressure. For our analysis, we need the following theorem (which is Theorem 1 in [9]).

Theorem 3: Suppose that γ_V^* is unique, that the slackness condition (2) holds, and that the dual function $g(\gamma)$ satisfies:

$$g(\gamma_V^*) \geq g(\gamma) + L\|\gamma_V^* - \gamma\| \quad \forall \gamma \succeq \mathbf{0}, \quad (12)$$

for some constant $L > 0$ independent of V . Then under Backpressure with FIFO or LIFO, there exist constants

²In [17], the theorem is proven under more general Markovian $S(t)$ processes that include the $S(t)$ process assumed here.

$D, K, c^* = \Theta(1)$, i.e., all independent of V , such that for any $m \in \mathbb{R}_+$,

$$\mathcal{P}^{(r)}(D, Km) \leq c^* e^{-m}, \quad (13)$$

where $\mathcal{P}^{(r)}(D, Km)$ is defined:

$$\mathcal{P}^{(r)}(D, Km) \quad (14)$$

$$\triangleq \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \Pr\{\exists j, |q_j(\tau) - \gamma_{Vj}^*| > D + Km\}.$$

Proof: See [9]. ■

Note that if a steady state distribution exists for $q(t)$, e.g., when all queue sizes are integers, then $\mathcal{P}^{(r)}(D, Km)$ is indeed the steady state probability that there exists a queue j whose queue value deviates from γ_{Vj}^* by more than $D + Km$ distance. In this case, Theorem 3 states that $q_j(t)$ deviates from γ_{Vj}^* by $\Theta(\log(V))$ distance with probability $O(1/V)$. Hence when V is large, $q_j(t)$ will mostly be within $O(\log(V))$ distance from γ_{Vj}^* . Also note that the conditions of Theorem 3 are not very restrictive. The condition (12) can usually be satisfied in practice when the action space is finite, in which case the dual function $g(\gamma)$ is polyhedral (see [9] for more discussions). The uniqueness of γ_V^* can usually be satisfied in many network utility optimization problems, e.g., [2].

We now present the main result of this paper with respect to the delay performance of LIFO-Backpressure. Below, the notion “average arrival rate” is defined as follows: Let $A_j(t)$ be the number of packets entering queue j at time t . Then the time average arrival rate of these packets is defined (assuming it exists): $\lambda_j = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} A_j(\tau)$. For the theorem, we assume that time averages under Backpressure exist with probability 1. This is a reasonable assumption, and holds whenever the resulting discrete time Markov chain for the queue vector $q(t)$ under backpressure is countably infinite and irreducible. Note that the state space is indeed countably infinite if we assume packets take integer units. If the system is also irreducible then the finite average backlog result of Theorem 2 implies that all states are positive recurrent.

Let D, K, c^* be constants as defined in Theorem 3, and recall that these are $\Theta(1)$ (independent of V). Assume $V \geq 1$, and define $Q_{j,High}$ and $Q_{j,Low}$ as:

$$\begin{aligned} Q_{j,High} &\triangleq \gamma_{Vj}^* + D + K[\log(V)]^2, \\ Q_{j,Low} &\triangleq \max[\gamma_{Vj}^* - D - K[\log(V)]^2, 0]. \end{aligned}$$

Define the interval $\mathcal{B}_j \triangleq [Q_{j,High}, Q_{j,Low}]$. The following theorem considers the rate and delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart.

Theorem 4: Suppose that $V \geq 1$, that γ_V^* is unique, that the slackness assumption (2) holds, and that the dual function $g(\gamma)$ satisfies:

$$g(\gamma_V^*) \geq g(\gamma) + L\|\gamma_V^* - \gamma\| \quad \forall \gamma \succeq \mathbf{0}, \quad (15)$$

for some constant $L > 0$ independent of V . Define D, K, c^* as in Theorem 3, and define \mathcal{B}_j as above. Then for any queue j with a time average input rate $\lambda_j > 0$, we have under LIFO-Backpressure that:

(a) The rate $\tilde{\lambda}_j$ of packets that both arrive to queue j when $q_j(t) \in \mathcal{B}_j$ and that eventually depart the queue satisfies:

$$\lambda_j \geq \tilde{\lambda}_j \geq \left[\lambda_j - \frac{\delta_{max} c^*}{V^{\log(V)}} \right]^+. \quad (16)$$

(b) The average delay of these packets is at most W_{bound} , where:

$$W_{bound} \triangleq [2D + 2K[\log(V)]^2 + \delta_{max}] / \tilde{\lambda}_j.$$

This theorem says that the delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart is at most $O([\log(V)]^2)$. Further, by (16), when V is large, these packets represent the overwhelming majority, in that the rate of packets not in this set is at most $O(1/V^{\log(V)})$.

Proof: (Theorem 4) Theorem 2 shows that average queue backlog is finite. Thus, there can be at most a finite number of packets that enter the queue and never depart, so the rate of packets arriving that never depart must be 0. It follows that $\tilde{\lambda}_j$ is equal to the rate at which packets arrive when $q_j(t) \in \mathcal{B}_j$. Define the indicator function $1_j(t)$ to be 1 if $q_j(t) \notin \mathcal{B}_j$, and 0 else. Define $\tilde{\lambda}_j^c \triangleq \lambda_j - \tilde{\lambda}_j$. Then with probability 1 we get:³

$$\begin{aligned} \tilde{\lambda}_j^c &= \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} A_j(\tau) 1_j(\tau) \\ &= \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{A_j(\tau) 1_j(\tau)\}. \end{aligned}$$

Then using the fact that $A_j(t) \leq \delta_{max}$ for all j, t :

$$\begin{aligned} \mathbb{E}\{A_j(t) 1_j(t)\} &= \mathbb{E}\{A_j(t) | q_j(t) \notin \mathcal{B}_j\} Pr\{q_j(t) \notin \mathcal{B}_j\} \\ &\leq \delta_{max} Pr\{q_j(t) \notin [Q_{j,Low}, Q_{j,High}]\}. \end{aligned}$$

Therefore:

$$\begin{aligned} \tilde{\lambda}_j^c &\leq \delta_{max} \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Pr(q_j(\tau) \notin [Q_{j,Low}, Q_{j,High}]) \\ &\leq \delta_{max} \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Pr(|q_j(\tau) - \gamma_{V,j}^*| > D + Km), \end{aligned}$$

where we define $m \triangleq [\log(V)]^2$, and note that $m \geq 0$ because $V \geq 1$. From Theorem 3 we thus have:

$$0 \leq \tilde{\lambda}_j^c \leq \delta_{max} c^* e^{-m} = \frac{\delta_{max} c^*}{V^{\log(V)}}. \quad (17)$$

This completes the proof of part (a). Now define $\tilde{\mathcal{B}}_j = [Q_{j,High} + \delta_{max}, Q_{j,Low}]$. Since $\mathcal{B}_j \subset \tilde{\mathcal{B}}_j$, we see that the rate of the packets that enter $\tilde{\mathcal{B}}_j$ is at least $\tilde{\lambda}_j$. Part (b) then follows from Theorem 1 and the facts that queue j is stable and that $|\tilde{\mathcal{B}}_j| \leq 2D + 2K[\log(V)]^2 + \delta_{max}$. ■

Note that if $\lambda_j = \Theta(1)$, we see from Theorem 4 that, under LIFO-Backpressure, the time average delay for almost all packets going through queue j is only $O([\log(V)]^2)$. Applying this argument to all network queues with $\Theta(1)$ input rates, we see that all but a tiny fraction of the traffic entering the network only experiences a delay of $O([\log(V)]^2)$. This contrasts with

the delay performance result of the usual Backpressure with FIFO, which states that the time average delay will be $\Theta(V)$ for all packets [9]. Also note that under LIFO-Backpressure, some packets may stay in the queue for very long time. This problem can be compensated by introducing certain coding techniques, e.g., fountain codes [18], into the LIFO-Backpressure algorithm.

VII. SIMULATION

In this section, we provide simulation results of the LIFO-Backpressure algorithm. We consider the network shown in Fig. 4, where we try to support a flow sourced by Node 1 destined for Node 7 with minimum energy consumption.

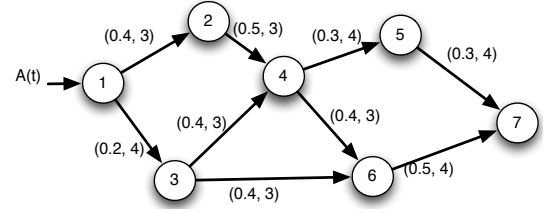


Fig. 4. A multihop network. (a, b) represents the *HIGH* probability a and the rate b obtained with one unit of power when *HIGH*.

We assume that $A(t)$ evolves according to the 2-state Markov chain in Fig. 5. When the state is *HIGH*, $A(t) = 3$, else $A(t) = 0$. We assume that the condition of each link can either be *HIGH* or *LOW* at a time. All the links except link (2, 4) and link (6, 7) are assumed to be i.i.d. every time slot, whereas the conditions of link (2, 4) and link (6, 7) are assumed to be evolving according to independent 2-state Markov chains in Fig. 5. Each link's *HIGH* probability and unit power rate at the *HIGH* state is shown in Fig. 4. The unit power rates of the links at the *LOW* state are all assumed to be 1. We assume that the link states are all independent and there is no interference. However, each node can only spend one unit of power per slot to transmit over one outgoing link, although it can simultaneously receive from multiple incoming links. The goal is to minimize the time average power while maintaining network stability.

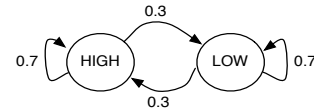


Fig. 5. The two state Markov chain with the transition probabilities.

We simulate Backpressure with both LIFO and FIFO for 10^6 slots with $V \in \{20, 50, 100, 200, 500\}$. It can be verified that the backlog vector converges to a unique attractor as V increases in this case. The left two plots in Fig. 6 show the average power consumption and the average backlog under LIFO-Backpressure. It can be observed that the average power quickly converges to the optimal value and that the average backlog grows linearly in V . The right plot of Fig. 6 shows the percentage of time when there exists a q_j whose value deviates from $\gamma_{V,j}^*$ by more than $2[\log(V)]^2$. As we can see,

³The time average expectation is the same as the pure time average by the Lebesgue Dominated Convergence Theorem, because we assume the pure time average exists with probability 1, and that $0 \leq A_j(t) \leq \delta_{max} \forall t$.

this percentage is always very small, i.e., between 0.002 and 0.013, showing a good match between the theory and the simulation results.

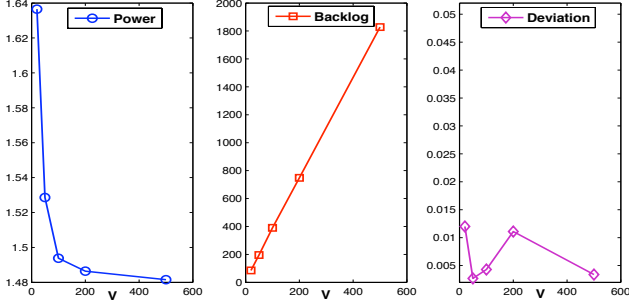


Fig. 6. LEFT: average network power consumption. MIDDLE: average network backlog size. RIGHT: percentage of time when $\exists q_j$ such that $|q_j - \gamma_{V,j}^*| > 2[\log(V)]^2$.

Fig. 7 compares the delay statistics of LIFO and FIFO for more than 99.9% of the packets that leave the system before the simulation ends, under the cases $V = 100$ and $V = 500$. We see that LIFO not only dramatically reduces the average packet delay for these packets, but also greatly reduces the delay for most of these packets. For instance, when $V = 500$, under FIFO, almost all packets experience the average delay around 1220 slots. Whereas under LIFO, the average packet delay is brought down to 78. Moreover, 52.9% of the packets only experience delay less than 20 slots, and 90.4% of the packets experience delay less than 100 slots. *Hence most packets' delay are reduced by a factor of 12 under LIFO as compared to that under FIFO!*

| V=100 | | | | |
|-------|---------|-----------|-----------|------------|
| Case | Avg. DL | % DL < 20 | % DL < 50 | % DL < 100 |
| LIFO | 55.4 | 55.0 | 82.1 | 91.8 |
| FIFO | 260.6 | 0 | 0 | 0 |
| V=500 | | | | |
| Case | Avg. DL | % DL < 20 | % DL < 50 | % DL < 100 |
| LIFO | 78.3 | 52.9 | 80.4 | 90.4 |
| FIFO | 1219.8 | 0 | 0 | 0 |

Fig. 7. Delay statistics under Backpressure with LIFO and FIFO for packets that leave the system before simulation ends (more than 99.9%). %DL < a is the percentage of packets that enter the network and has delay less than a.

Fig. 8 also shows the delay for the first 20000 packets that enter the network in the case when $V = 500$. We see that under Backpressure plus LIFO, most of the packets experience very small delay; while under Backpressure with FIFO, each packet experiences roughly the average delay.

VIII. EMPIRICAL VALIDATION

In this section we validate our analysis empirically by carrying out new experiments over the same testbed and Backpressure Collection Protocol (BCP) code of [10]. This prior work did not empirically evaluate the relationship between V , finite storage availability, packet latency and packet discard rate. We note that BCP runs atop the default CSMA MAC for TinyOS which is not known to be throughput optimal, that the testbed may not precisely be defined by a finite state Markovian

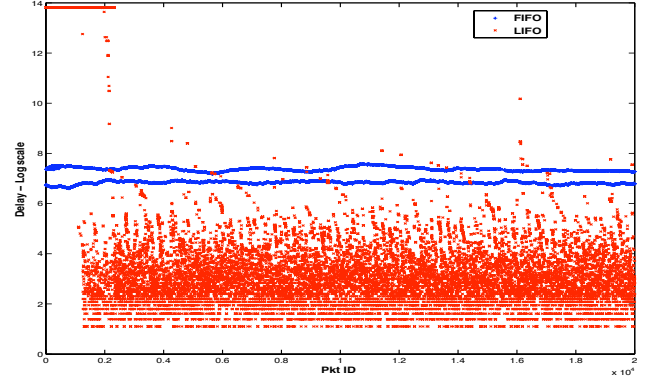


Fig. 8. Packet Delay under Backpressure with LIFO and FIFO

evolution, and finally that limited storage availability on real wireless sensor nodes mandates the introduction of virtual queues to maintain backpressure values in the presence of data queue overflows.

In order to avoid using very large data buffers, in [10] the forwarding queue of BCP has been implemented as a *floating queue*. The concept of a floating queue is shown in Figure 10, which operates with a finite data queue of size D_{max} residing atop a virtual queue which preserves backpressure levels. Packets that arrive to a full data queue result in a data queue discard and the incrementing of the underlying virtual queue counter. Underflow events (in which a virtual backlog exists but the data queue is empty) results in null packet generation, which are filtered and then discarded by the destination.

Despite these real-world differences, we are able to demonstrate clear order-equivalent delay gains due to LIFO usage in BCP in the following experimentation.

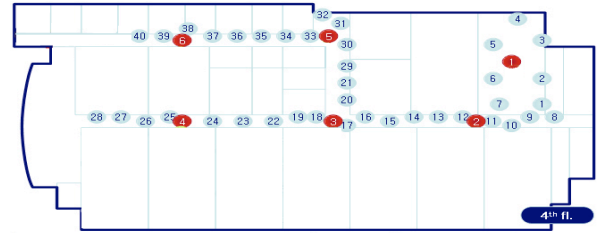


Fig. 9. The 40 tMote Sky devices used in experimentation on Tutornet.

A. Testbed and General Setup

To demonstrate the empirical results, we deployed a collection scenario across 40 nodes within the Tutornet testbed (see Figure 9). This deployment consisted of Tmote Sky devices embedded in the 4th floor of Ronald Tutor Hall at the University of Southern California.

In these experiments, one sink mote (ID 1 in Figure 9) was designated and the remaining 39 motes sourced traffic simultaneously, to be collected at the sink. The Tmote Sky devices were programmed to operate on 802.15.4 channel 26, selected for the low external interference in this spectrum on Tutornet. Further, the motes were programmed to transmit at -15 dBm to provide reasonable interconnectivity. These experimental settings are identical to those used in [10].

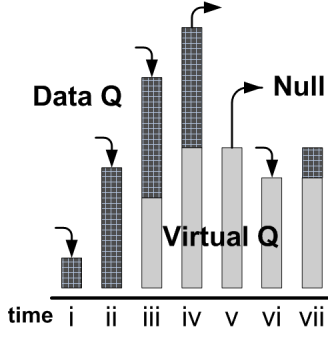


Fig. 10. The floating LIFO queues of [10] drop from the data queue during overflow, placing the discards within an underlying virtual queue. Services that cause data queue underflows generate null packets, reducing the virtual queue size.

We vary D_{max} over experimentation. In practice, BCP defaults to a D_{max} setting of 12 packets, the maximum reasonable resource allocation for a packet forwarding queue in these highly constrained devices.

B. Experiment Parameters

Experiments consisted of Poisson traffic at 1.0 packets per second per source for a duration of 20 minutes. This source load is moderately high, as the boundary of the capacity region for BCP running on this subset of nodes has previously been documented at 1.6 packets per second per source [10]. A total of 36 experiments were run using the standard BCP LIFO queue mechanism, for all combinations of $V \in \{1, 2, 3, 4, 6, 8, 10, 12\}$ and LIFO storage threshold $D_{max} \in \{2, 4, 8, 12\}$. In order to present a delay baseline for Backpressure we additionally modified the BCP source code and ran experiments with 32-packet FIFO queues (no floating queues) for $V \in \{1, 2, 3\}$.⁴

C. Results

Testbed results in Figure 11 provide the system average packet delay from source to sink over V and D_{max} , and includes 95% confidence intervals. Delay in our FIFO implementation scales linearly with V , as predicted by the analysis in [9]. This yields an average delay that grows very rapidly with V , already greater than 9 seconds per packet for $V = 3$. Meanwhile, the LIFO floating queue of BCP performs much differently. We have plotted a scaled $[\log(V)]^2$ target, and note that as D_{max} increases the average packet delay remains bounded by $\Theta([\log(V)]^2)$.

These delay gains are only possible as a result of discards made by the LIFO floating queue mechanism that occur when the queue size fluctuates beyond the capability of the finite data queue to smooth. Figure 12 gives the system packet loss rate of BCP's LIFO floating queue mechanism over V . Note that the poly-logarithmic delay performance of Figure 11 is achieved even for data queue size 12, which itself

⁴These relatively small V values are due to the constraint that the nodes have small data buffers. Using larger V values will cause buffer overflow at the nodes.

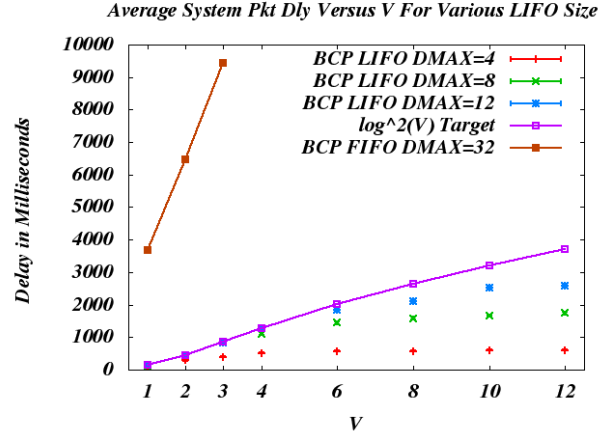


Fig. 11. System average source to sink packet delay for BCP FIFO versus BCP LIFO implementation over various V parameter settings.

drops at most 5% of traffic at $V = 12$. We cannot state conclusively from these results that the drop rate scales like $O(\frac{1}{V \log(V)})$. We hypothesize that a larger V value would be required in order to observe the predicted drop rate scaling. Bringing these results back to real-world implications, note that BCP (which minimizes a penalty function of packet retransmissions) performs very poorly with $V = 0$, and was found to have minimal penalty improvement for V greater than 2. At this low V value, BCP's 12-packet forwarding queue demonstrates zero packet drops in the results presented here. These experiments, combined with those of [10] suggest strongly that the drop rate scaling may be inconsequential in many real world applications.

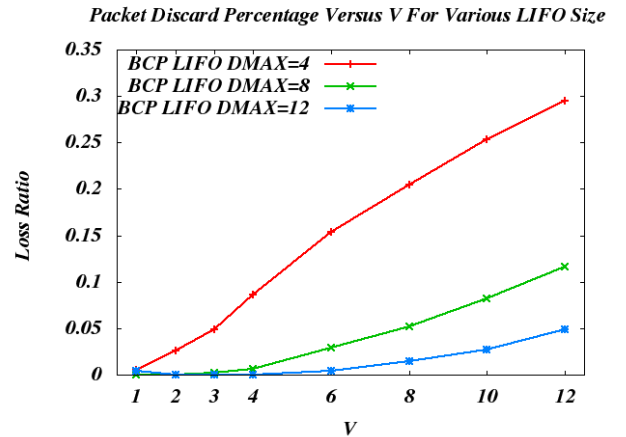


Fig. 12. System packet loss rate of BCP LIFO implementation over various V parameter settings.

In order to explore the queue backlog characteristics and compare with our analysis, Figure 13 presents a histogram of queue backlog frequency for rear-network-node 38 over various V settings. This node was observed to have the worst queue size fluctuations among all thirty-nine sources. For $V = 2$, the queue backlog is very sharply distributed and fluctuates outside the range $[11 - 15]$ only 5.92% of the

experiment. As V is increased, the queue attraction is evident. For $V = 8$ we find that the queue deviates outside the range $[41 - 54]$ only 5.41% of the experiment. The queue deviation is clearly scaling sub-linearly, as a four-fold increase in V required only a 2.8 fold increase in D_{max} for comparable drop performance.

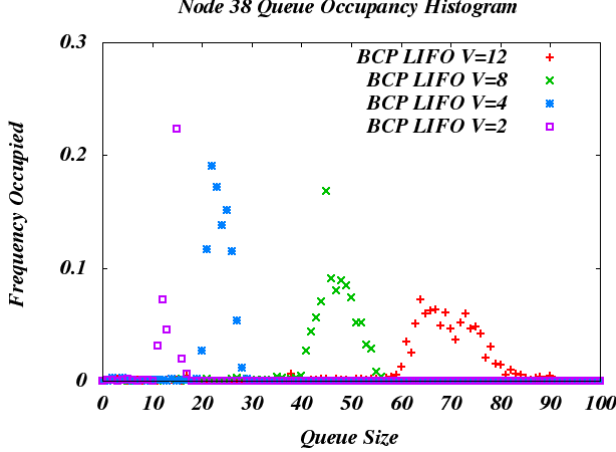


Fig. 13. Histogram of queue backlog frequency for rear-network-node 38 over various V settings.

IX. OPTIMIZING FUNCTIONS OF TIME AVERAGES

So far we have focused on optimizing time averages of functions, we now consider the case when the objective of the network controller is to optimize a function of some time average metric, e.g., [15]. Specifically, we assume that the action $x(t)$ at time t incurs some instantaneous *network attribute vector* $\mathbf{y}(t) = \mathbf{y}(x(t)) = (y_1(t), \dots, y^K(t))^T \in \mathbb{R}_+^K$, and the objective of the network controller is to minimize a cost function $\text{Cost}(\overline{\mathbf{y}(t)}) : \mathbb{R}_+^K \rightarrow \mathbb{R}_+$,⁵ where $\overline{\mathbf{y}(t)}$ represents the time average value of $\mathbf{y}(t)$. We assume that the function $\text{Cost}(\cdot)$ is continuous, convex and is component-wise increasing, and that $|y_k(x(t))| \leq \delta_{max}$ for all $k, x(t)$. In this case, we see that the Backpressure algorithm in Section V cannot be directly applied and the deterministic problem (8) also needs to be modified.

To tackle this problem using the Backpressure algorithm, we introduce an *auxiliary vector* $\mathbf{z}(t) = (z_1(t), \dots, z_K(t))^T$. We then define the following virtual queues $H_k(t), j = 1, \dots, K$ that evolves as follows:

$$H_k(t+1) = \max [H_k(t) - y_k(t), 0] + z_k(t). \quad (18)$$

These virtual queues are introduced for ensuring that the average value of $y_k(t)$ is no less than the average value of $z_k(t)$. We will then try to maximize the time average of the function $\text{Cost}(\mathbf{z}(t))$, subject to the constraint that the actual queues $q_j(t)$ and the virtual queues $H_k(t)$ must all be stable. Specifically, the Backpressure algorithm for this problem works as follows:

⁵The case for maximizing a utility function of long term averages can be treated in a similar way.

Backpressure: At every time slot t , observe the current network state $S(t)$, and the backlogs $\mathbf{q}(t)$ and $\mathbf{H}(t)$. If $S(t) = s_i$, do the following:

- 1) **Auxiliary vector:** choose the vector $\mathbf{z}(t)$ by solving:

$$\begin{aligned} \min : \quad & V\text{Cost}(\mathbf{z}) + \sum_k H_k(t)z_k \\ \text{s.t.} \quad & 0 \leq z_k \leq \delta_{max}. \end{aligned} \quad (19)$$

- 2) **Action:** choose the action $x(t) \in \mathcal{X}^{(s_i)}$ that solves:

$$\begin{aligned} \max : \quad & \sum_k H_k(t)y_k(x) + \sum_j q_j(t)[\mu_j(s_i, x) - A_j(s_i, x)] \\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \end{aligned} \quad (20)$$

In this case, one can also show that this Backpressure algorithm achieves the $[O(1/V), O(V)]$ utility-delay tradeoff under a Markovian $S(t)$ process. We also note that in this case, the deterministic problem is slightly different. Indeed, the intuitively formulation will be of the following form:

$$\begin{aligned} \min : \quad & \mathcal{F}(\mathbf{x}) \triangleq V\text{Cost}\left(\sum_{s_i} \pi_{s_i} \mathbf{y}(x^{(s_i)})\right) \\ \text{s.t.} \quad & \mathcal{A}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \\ & \leq \mathcal{B}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \quad \forall j \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall i = 1, 2, \dots, M. \end{aligned} \quad (21)$$

However, the dual problem of this optimization problem is not separable, i.e., not of the form of (10), unless the function $\text{Cost}(\cdot)$ is linear or if there exists an optimal action that is in every feasible action set $\mathcal{X}^{(s_i)}$, e.g., [15]. To get rid of this problem, we introduce the auxiliary vector $\mathbf{z} = (z_1, \dots, z_K)^T$ and change the problem to:

$$\begin{aligned} \min : \quad & \mathcal{F}(\mathbf{x}) \triangleq V\text{Cost}(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{z} \preceq \sum_{s_i} \pi_{s_i} \mathbf{y}(x^{(s_i)}) \\ & \mathcal{A}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \\ & \leq \mathcal{B}_j(\mathbf{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \quad \forall j \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall i = 1, 2, \dots, M. \end{aligned} \quad (22)$$

It can be shown that this modified problem is equivalent to (21). Now we see that it is indeed due to the non-separable feature of (21) that we need to introduce the auxiliary vector $\mathbf{z}(t)$ in the Backpressure problem. We also note that the problem (22) actually has the form of (8). Therefore, all previous results on (8), e.g., Theorem 3 and 4 will also apply to problem (22).

APPENDIX A – PROOF OF 1

Here we provide the proof of Theorem 1.

Proof: Consider a sample path for which the lim inf arrival rate is at least λ_{min} and for which we have an infinite number of departures (this happens with probability 1 by assumption). There must be a non-empty subset of \mathcal{B} consisting of buffer locations that experience an infinite

number of departures. Call this subset $\tilde{\mathcal{B}}$. Now let $W_i^{(b)}$ be the delay of the i^{th} departure from b , let $D^{(b)}(t)$ denote the number of departures from a buffer slot $b \in \tilde{\mathcal{B}}$ up to time t , and use $Q^{(b)}(t)$ to denote the occupancy of the buffer slot b at time t . Note that $Q^{(b)}(t)$ is either 0 or 1. For all $t \geq 0$, it can be shown that:

$$\sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} \leq \int_0^t Q^{(b)}(\tau) d\tau. \quad (23)$$

This can be seen from Fig. 14 below.

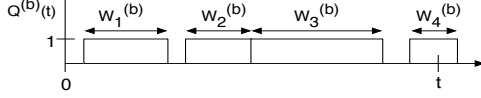


Fig. 14. An illustration of inequality (23) for a particular buffer location b . At time t in the figure, we have $D^{(b)}(t) = 3$.

Therefore, we have:

$$\begin{aligned} \sum_{b \in \tilde{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} &\leq \int_0^t \sum_{b \in \tilde{\mathcal{B}}} Q^{(b)}(\tau) d\tau \\ &\leq \int_0^t |\tilde{\mathcal{B}}| d\tau \\ &\leq |\mathcal{B}|t. \end{aligned} \quad (24)$$

The left-hand-side of the above inequality is equal to the sum of all delays of jobs that depart from locations in $\tilde{\mathcal{B}}$ up to time t . All other buffer locations (in \mathcal{B} but not in $\tilde{\mathcal{B}}$) experience only a finite number of departures. Let \mathcal{J} represent an index set that indexes all of the (finite number) of jobs that depart from these other locations. Note that the delay W_j for each job $j \in \mathcal{J}$ is finite (because, by definition, job j eventually departs). We thus have:

$$\sum_{i=1}^{D(t)} W_i \leq \sum_{b \in \tilde{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} + \sum_{j \in \mathcal{J}} W_j.$$

where the inequality is because the second term on the right-hand-side sums over jobs in \mathcal{J} , and these jobs may not have departed by time t . Combining the above and (24) yields for all $t \geq 0$:

$$\sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}|t + \sum_{j \in \mathcal{J}} W_j.$$

Dividing by $D(t)$ yields:

$$\frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq \frac{|\mathcal{B}|t}{D(t)} + \frac{1}{D(t)} \sum_{j \in \mathcal{J}} W_j.$$

Taking a lim sup as $t \rightarrow \infty$ yields:

$$\limsup_{t \rightarrow \infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq \limsup_{t \rightarrow \infty} \frac{|\mathcal{B}|t}{D(t)}, \quad (25)$$

where we have used the fact that $\sum_{j \in \mathcal{J}} W_j$ is a finite number, and $D(t) \rightarrow \infty$ as $t \rightarrow \infty$, so that:

$$\limsup_{t \rightarrow \infty} \frac{1}{D(t)} \sum_{j \in \mathcal{J}} W_j = 0.$$

Now note that, because each buffer location in \mathcal{B} can hold at most one job, the number of departures $D(t)$ is at least $N(t) - |\mathcal{B}|$, which is a positive number for sufficiently large t . Thus:

$$\begin{aligned} \limsup_{t \rightarrow \infty} \frac{|\mathcal{B}|t}{D(t)} &\leq \limsup_{t \rightarrow \infty} \left[\frac{|\mathcal{B}|t}{N(t) - |\mathcal{B}|} \right] \\ &= \limsup_{t \rightarrow \infty} \left[\frac{|\mathcal{B}|}{N(t)/t - |\mathcal{B}|/t} \right] \\ &\leq |\mathcal{B}|/\lambda_{\min}. \end{aligned}$$

Using this in (25) proves the result. \blacksquare

REFERENCES

- [1] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control*, vol. 37, no. 12, pp. 1936-1949, Dec. 1992.
- [2] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333-1344, 2007.
- [3] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *Proc. of 45th Annual Allerton Conference on Communication, Control, and Computing (invited paper)*, Sept. 2007.
- [4] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *IEEE INFOCOM Proceedings*, April 2008.
- [5] Y. Yi and M. Chiang. Stochastic network utility maximization: A tribute to Kelly's paper published in this journal a decade ago. *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 421-442, June 2008.
- [6] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.
- [7] M. J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Nonlinear Optimization of Communication Systems*, 24(8), Aug. 2006.
- [8] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *IEEE Transactions on Information Theory* vol. 53, no. 9, pp. 3095-3113, Sept. 2007.
- [9] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Transactions on Automatic Control*, to appear.
- [10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.
- [11] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar. Backpressure-based packet-by-packet adaptive routing in communication networks. *arXiv:1005.4984v1*, May 2010.
- [12] L. Bui, R. Srikant, and A. Stolyar. Optimal resource allocation for multicast sessions in multi-hop wireless networks. *Philosophical Transactions of The Royal Society, Series A*, pages 2059-2074, Jan 2008.
- [13] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM 2009 Mini-Conference*, April 2009.
- [14] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice Hall, 1992.
- [15] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE INFOCOM Proceedings*, March 2005.
- [16] M. J. Neely. Stability and capacity regions for discrete time queueing networks. *arXiv:1003.3396v1*, March 2010.
- [17] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.
- [18] M. Mitzenmacher. Digital fountains: A survey and look forward. *IEEE Information Theory Workshop (ITW), San Antonio, Texas*, Oct. 2004.